Тема
«Лексика языка. Типы данных.»

Состав

Переменные и типы данных

Переменная — это именованная область памяти для хранения данных, которые могут изменяться в процессе исполнения программы. Переменная характеризуется: Именем («обозначением ячейки памяти») и Значением (данными, содержащимися в переменной в конкретный момент времени)

В Java существует две группы типов данных:
- Примитивные;
- Ссылочные (объектные).

Примитивные типы данных:
- **byte** – 8-битное знаковое целое число. Диапазон значений: от -128 до 127.
  Пример объявления переменной: `byte a = 3;`
- **short** – 16-битное знаковое целое число. Диапазон значений: от -32768 до 32767.
  Пример объявления переменной: `short a = 2019;`
- int – 32-битное знаковое целое число. Диапазон значений: от -2147483648 до 2147483647.
  Пример объявления переменной: `int a = 200;`
- **long** – 64-битное знаковое целое число. Диапазон значений: от -9223372036854775808 до 9223372036854775807.
  Пример объявления переменной: `long a = 3000L;` (после числа ставится буква L)
- **float** – 32-битное знаковое число с плавающей запятой одинарной точности.
  Пример объявления переменной: `float a = 120.0f;` (после числа ставится буква f)
- **double** – 64-битное знаковое число с плавающей запятой двойной точности.
  Пример объявления переменной: `double a = 93.1123;`
- **boolean** – принимает два значения: true и false.
  Пример объявления переменной: `boolean a = true;`
- **char** – 16-битный тип данных, предназначенный для хранения символов в кодировке Unicode. Диапазон значений: от '\u0000' или 0 до '\uffff' или 65,535.
  Пример объявления переменной: `char c = 'X';`

Общую структуру объявления переменной можно описать как:
[тип_данных] [идентификатор(имя_переменной)] = [начальное_значение];

Например:

```
int a = 37;
double b;
b = 2.25;
```

Имена, использующиеся для переменных, функций, меток и других определяемых пользователем объектов, называются **идентификаторами**.
Идентификаторы в Java должны начинаться с буквы, $ или _, а затем может идти любая последовательность символов. Идентификаторы чувствительны к регистру. Ключевые слова Java не могут быть идентификаторами.

В Java зарезервированы следующие ключевые слова:
abstract, assert, boolean, break, byte, case, catch, char, class, const, continue, default, do, double, else, enum, extends, final, finally, float, for, goto, if, implements, import, instanceof, int, interface, long, native, new, package, protected, public, return, short, static, strictfp, super, switch, synchronized, this, throw, throws, transient, try, void, volatile, while

Переменные, которые не могут менять свои значения называют константами. Для объявления переменной как константы в Java используется ключевое слово final. Пример объявления константы: `final int a = 20;`

Арифметические операции

Большинство операций в Java аналогичны тем, которые применяются в других си-подобных языках. Есть унарные операции (выполняются над одним операндом), бинарные - над двумя операндами, а также тернарные - выполняются над тремя операндами. Операндом является переменная или значение (например, число), участвующее в операции. Рассмотрим все виды операций.

В арифметических операциях участвуют числами. В Java есть бинарные арифметические операции (производятся над двумя операндами) и унарные (выполняются над одним операндом). К бинарным операциям относят следующие:

- операция сложения двух чисел:
```
int a = 3, b = 2;
int c = a + b; // 5
```
- операция вычитания двух чисел:
```
int a = 3, b = 2;
int c = a - b; // 1
```
- операция умножения двух чисел:
```
int a = 3, b = 2;
int c = a * b; // 6
```
- операция деления двух чисел:
```
int a = 3, b = 2;
int c = a / b; // 1
double d = 22.5 / 4.5; // 5.0
```
При делении стоит учитывать, что если в операции участвуют два целых числа, то результат деления будет округляться до целого числа, даже если результат присваивается переменной float или double.
- получение остатка от деления двух чисел:
```
int a = 3, b = 2;
int c = a % b; // 1
```
- ++ (префиксный инкремент):
```
int a = 3;
int b = ++a; // 4
```
- ++ (постфиксный инкремент):
```
int a = 3;
int b = a++; // b = 3, a = 4;
```
- -- (префиксный декремент):
```
int a = 3;
int b = --a; // 2
```
- -- (постфиксный декремент):
```
int a = 3;
int b = a--; // b = 3, a = 2
```

Приоритет арифметических операций

Одни операции имеют больший приоритет чем другие и поэтому выполняются вначале. Операции в порядке уменьшения приоритета:

++ (инкремент), -- (декремент)
* (умножение), / (деление), % (остаток от деления)
+ (сложение), - (вычитание)

Методы

Если переменные и константы хранят некоторые значения, то методы содержат собой набор операторов, которые выполняют определенные действия.

Общее определение методов выглядит следующим образом:

```
[модификаторы] тип_возвращаемого_значения название_метода ([параметры]){
    // тело метода
}
```

Модификаторы и параметры необязательны.

По умолчанию главный класс любой программы на Java содержит метод main, который служит точкой входа в программу:

```
public static void main(String[] args) {
    System.out.println("привет мир!");
}
```

Ключевые слова public и static являются модификаторами. Далее идет тип возвращаемого значения. Ключевое слово void указывает на то, что метод ничего не возвращает.

Затем идут название метода - main и в скобках параметры метода - String[] args. И в фигурные скобки заключено тело метода - все действия, которые он выполняет.

Условные операторы

Одним из фундаментальных элементов многих языков программирования являются условные конструкции. Данные конструкции позволяют направить работу программы по одному из путей в зависимости от определенных условий.

В языке Java используются следующие условные конструкции: if..else и switch..case

Конструкция if/else

Выражение if/else проверяет истинность некоторого условия и в зависимости от результатов проверки выполняет определенный код:

```
int num1 = 6;
int num2 = 4;
if(num1 > num2){
    System.out.println("Первое число больше второго");
}
```

После ключевого слова if ставится условие. И если это условие выполняется, то срабатывает код, который помещен в далее в блоке if после фигурных скобок. В качестве условий выступает операция сравнения двух чисел.

Так как, в данном случае первое число больше второго, то выражение num1 > num2 истинно и возвращает значение true. Следовательно, управление переходит в блок кода после фигурных скобок и начинает выполнять содержащиеся там инструкции, а конкретно метод System.out.println("Первое число больше второго");. Если бы первое число оказалось бы меньше второго или равно ему, то инструкции в блоке if не выполнялись бы.

Но что, если мы захотим, чтобы при несоблюдении условия также выполнялись какие-либо действия? В этом случае мы можем добавить блок else:

```
int num1 = 6;
int num2 = 4;
if(num1 > num2) {
    System.out.println("Первое число больше второго");
} else {
    System.out.println("Первое число меньше второго");
}
```

Но при сравнении чисел мы можем насчитать три состояния: первое число больше второго, первое число меньше второго и числа равны. С помощью выражения else if, мы можем обрабатывать дополнительные условия:

```
int num1 = 6;
int num2 = 8;
if(num1>num2) {
    System.out.println("Первое число больше второго");
} else if(num1 <num2) {
    System.out.println("Первое число меньше второго");
} else {
    System.out.println("Числа равны");
}
```

Также мы можем соединить сразу несколько условий, используя логические операторы:

```
int num1 = 8;
int num2 = 6;
if(num1 > num2 && num1>7){
    System.out.println("Первое число больше второго и больше 7");
}
```

Здесь блок if будет выполняться, если num1 > num2 равно true и одновременно num1>7 равно true.

Конструкция switch

Конструкция switch/case аналогична конструкции if/else, так как позволяет обработать сразу несколько условий:

```
int num = 8;
switch(num){

    case 1:
        System.out.println("число равно 1");
        break;
    case 8:
        System.out.println("число равно 8");
        num++;
        break;
    case 9:
        System.out.println("число равно 9");
        break;
    default:
        System.out.println("число не равно 1, 8, 9");
}
```

После ключевого слова switch в скобках идет сравниваемое выражение. Значение этого выражения последовательно сравнивается со значениями, помещенными после оператора case. И если совпадение будет найдено, то будет выполняться определенный блок case.

В конце блока case ставится оператор break, чтобы избежать выполнения других блоков. Например, если бы убрали бы оператор break в следующем случае:

```
case 8:
    System.out.println("число равно 8");
    num++;
case 9:
    System.out.println("число равно 9");
    break;
```

то так как у нас переменная num равно 8, то выполнился бы блок case 8, но так как в этом блоке переменная num увеличивается на единицу, оператор break отсутствует, то начал бы выполняться блок case 9.

Если мы хотим также обработать ситуацию, когда совпадения не будет найдено, то можно добавить блок default, как в примере выше. Хотя блок default необязателен.

Также мы можем определить одно действие сразу для нескольких блоков case подряд:

```
int num = 3;
int output = 0;
switch(num){

    case 1:
        output = 3;
        break;
    case 2:
    case 3:
    case 4:
        output = 6;
        break;
    case 5:
```

```
        output = 12;
        break;
    default:
        output = 24;
}
System.out.println(output);
```

Тернарная операция
Тернарную операция имеет следующий синтаксис: [первый операнд - условие] ? [второй операнд] : [третий операнд]. Таким образом, в этой операции участвуют сразу три операнда. В зависимости от условия тернарная операция возвращает второй или третий операнд: если условие равно true, то возвращается второй операнд; если условие равно false, то третий. Например:

```
int x = 3;
int y = 2;
int z = x < y ? (x + y) : (x - y);
System.out.println(z);
```

Здесь результатом тернарной операции является переменная z. Сначала проверяется условие x<y. И если оно соблюдается, то z будет равно второму операнду - (x+y), иначе z будет равно третьему операнду.